

On Computation Rates for Arithmetic Sum

Ardhendu Tripathy and Aditya Ramamoorthy

Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011–3060

Email: ardhendu@iastate.edu, adityar@iastate.edu

Abstract—For zero-error function computation over directed acyclic networks, existing upper and lower bounds on the computation capacity are known to be loose. In this work we consider the problem of computing the arithmetic sum over a specific directed acyclic network that is not a tree. We assume the sources to be i.i.d. Bernoulli with parameter $1/2$. Even in this simple setting, we demonstrate that upper bounding the computation rate is quite nontrivial. In particular, it requires us to consider variable length network codes and relate the upper bound to equivalently lower bounding the entropy of descriptions observed by the terminal conditioned on the function value. This lower bound is obtained by further lower bounding the entropy of a so-called *clumpy distribution*. We also demonstrate an achievable scheme that uses variable length network codes and in-network compression.

I. INTRODUCTION

We consider the problem of function computation [1]–[4] using network coding [5], [6]. The setup of the problem typically is as follows. A directed acyclic graph (DAG) with capacity constraints is used to model a communication network. Certain nodes in the graph, referred to as terminals are interested in computing a target function of the data observed at some nodes (called sources) in the graph. The edges model error-free communication links and the nodes are assumed to be able to perform network coding. The objective is to find the maximum rate at which a network code can enable the terminals to compute estimates of the target function within a specified level of distortion.

The problem in its most general setting is known to be hard, and that has prompted the study of certain special cases [2], [7]–[9]. The specific case when the network has one terminal and the function needs to be computed without any distortion has received significant attention. Under this setup, one can consider either the zero-error setting or a setting where ϵ -error (for arbitrary $\epsilon > 0$) error is allowed. In [2], the authors do not assume a joint probability distribution on the input data and focus on zero-error function computation. After describing the amount of information that needs to be transmitted for this case, the same concept is applied to cuts that separate one or more sources from the terminal in a function computation problem over a DAG. Using this approach they were able to characterize the maximum achievable computation rate and network codes that obtain it for multi-edge tree networks. The work done in [10] approaches the function computation problem in a slightly different manner. Rather than focus on a computation rate for the entire network, they look at the rate region obtained by the vector of achievable rates over each edge in the network. They consider two scenarios:

worst case and *average case* complexity. The worst case complexity is related to the setting in [2], while the average case complexity assumes a joint probability distribution on the input data. For both scenarios, they use cut-set based arguments to characterize the rate region for tree networks. For general DAGs, cut-set based upper bounds are shown in [11] to be loose.

In this work we examine zero-error arithmetic sum computation over a specific DAG, but we assume a probability distribution on the input data (see Fig. 1). Note that for such a network there are two distinct paths from the source s_3 to the terminal that allows for multitude of network coding options. In [2], the same network was considered in a zero-error setting, but they did not assume any distribution on the inputs. They demonstrated an upper bound on the computation rate and a matching achievable scheme. In general, the distribution of the inputs is important in defining an associated rate of computation. Indeed, if the source values are deterministic, then the actual computation does not require any information to be transmitted on the edges.

In this work, we assume that each of the source is distributed i.i.d. Bernoulli with parameter $1/2$. We demonstrate that in this setting, upper bounding the computation rate is significantly harder because of the possibility of compressing the intermediate transmissions. In addition, the presence of multiple paths for source s_3 allows for many ways in which the information transfer and compression can be performed. Our upper bounds on the computation rate stem from the study of the entropy of the distribution of the descriptions transmitted by nodes s_1 and s_2 conditioned on the value of the arithmetic sum. Indeed, note that the arithmetic sum of two i.i.d. Bernoulli random variables has a biased probability mass function and one can lower transmission rates by appropriately compressing these values. For zero-error compression in the single source setting, it is well recognized that variable length codes are needed. Accordingly, we consider the class of variable length network codes that allow for computation of the arithmetic sum.

A. Main contribution

- We consider a variable-length network code for arithmetic sum computation in the particular DAG shown in Figure 1. In this variable length setting, we present an upper bound and a lower bound for the computation rate. The upper bound arises from studying the entropy of the descriptions communicated by s_1 and s_2 to t conditioned on the value of the sum. We show that this conditional

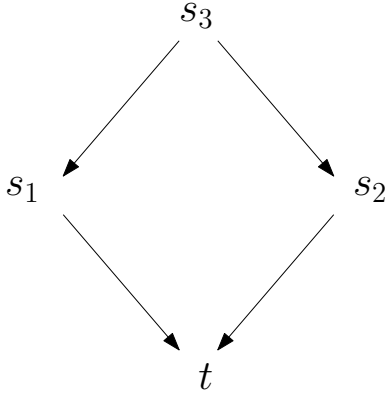


Fig. 1. A directed acyclic network with three sources and one terminal.

entropy can be lower bounded by an appropriately characterized “clumpy” distribution. The lower bound uses variable length codes for compression.

This paper is organized as follows. Section II presents the problem formulation. Section III uses a lower bound on the conditional entropy of the descriptions transmitted that is derived in Section IV to give an upper bound on the computation rate. Section V discusses an achievable scheme and Section VI concludes the paper.

II. PROBLEM FORMULATION

The edges in Figure 1 (later denoted by an ordered pair of vertices) have unit-capacity. Suppose that \mathcal{Z} is the alphabet used for communication, and $\mathcal{Z} > 1$. s_1, s_2, s_3 are the three source nodes that observe independent uniform iid sources X_1, X_2, X_3 respectively, each from $\{0, 1\}$. Terminal node t wants to compute the arithmetic sum $\Sigma = X_1 + X_2 + X_3$, $\Sigma \in \{0, 1, 2, 3\}$. WLOG we assume that edges $(s_3, s_1), (s_3, s_2)$ forward the value of X_3 to s_1, s_2 respectively. We adapt a variable-length network code to this function computation problem. In what follows, all logarithms denoted as \log are to the base 2 unless specified otherwise.

Definition 1: Let \mathcal{Z}^* denote the set of all finite-length sequences with alphabet \mathcal{Z} . A variable-length (k, N) network code for the network in Figure 1 has the following components.

- 1) Encoding functions for the edges, $e \in \{(s_1, t), (s_2, t)\}$:

$$\phi_e : \{0, 1\}^k \times \{0, 1\}^k \rightarrow \mathcal{Z}^*$$

Let $\mathbf{Z}_1 := \phi_{(s_1, t)}(\mathbf{X}_1, \mathbf{X}_3)$, $\mathbf{Z}_2 := \phi_{(s_2, t)}(\mathbf{X}_2, \mathbf{X}_3)$ where \mathbf{X}_j denotes a k -length random vector. $\mathbf{X}_j(i)$ refers to the i -th component of \mathbf{X}_j . We also let \mathbf{X}_j^n represent the vector $(\mathbf{X}_j(1), \dots, \mathbf{X}_j(n))$.

- 2) Decoding function for the terminal t :

$$\psi_t : \mathcal{Z}^N \times \mathcal{Z}^N \rightarrow \{0, 1, 2, 3\}^k$$

where random variable N is a *stopping time* with respect to the sequence $(\mathbf{Z}_1(1), \mathbf{Z}_2(1)), (\mathbf{Z}_1(2), \mathbf{Z}_2(2)), \dots$. Thus, the indicator function $\mathbf{1}_{\{N=n\}}$ is a function of $((\mathbf{Z}_1(1), \mathbf{Z}_2(1)), \dots, (\mathbf{Z}_1(n), \mathbf{Z}_2(n)))$.

Terminal t estimates the k -length component-wise arithmetic sum (denoted as $\hat{\Sigma}$) by setting $\hat{\Sigma} := \psi_t(\mathbf{Z}_1^N, \mathbf{Z}_2^N)$.

Definition 2: We say that a (k, N) network code *recovers* Σ with zero error if $\Pr(\hat{\Sigma} \neq \Sigma) = 0$, for all $\Sigma \in \{0, 1, 2, 3\}^k$. The rate of such a network code is defined as $\frac{k}{\mathbb{E}N \log |\mathcal{Z}|}$, where $\mathbb{E}N$ denotes the expected value of N . The capacity is

$$\mathcal{C} := \sup \left\{ \frac{k}{\mathbb{E}N \log |\mathcal{Z}|} : \begin{array}{l} \text{there is a zero-error } (k, N) \\ \text{network code that recovers } \Sigma. \end{array} \right\}$$

It can be observed that each component $1 \leq i \leq k$ of Σ is independent and identically distributed as follows.

$$\Pr(\Sigma(i) = \beta) = \begin{cases} 1/8, & \text{if } \beta \in \{0, 3\}, \\ 3/8, & \text{if } \beta \in \{1, 2\}. \end{cases}$$

In this work we derive upper and lower bounds on \mathcal{C} for the network in Fig. 1.

III. UPPER BOUND ON COMPUTING CAPACITY

Based on the relationships between the various quantities defined, we have the following.

$$\begin{aligned} H(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N) &= H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | N) + H(N), \\ &= H(N) + \sum \Pr(N = n) H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | N = n), \\ &\leq H(N) + 2 \log |\mathcal{Z}| \mathbb{E}N. \end{aligned}$$

The last inequality above is due to the fact that $H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | N = n) \leq 2n \log |\mathcal{Z}|$ bits. Furthermore,

$$\begin{aligned} &H(N) + 2 \log |\mathcal{Z}| \mathbb{E}N \\ &\geq H(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N | \Sigma) + I(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N; \Sigma), \\ &= H(\Sigma) - H(\Sigma | \mathbf{Z}_1^N, \mathbf{Z}_2^N, N) + H(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N | \Sigma), \\ &= H(\Sigma) + H(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N | \Sigma). \end{aligned}$$

The last equality above is true by the zero-error criterion. Note that the probability mass function for Σ implies that $H(\Sigma) = 1.8113k$ bits. Thus, we have that

$$H(\mathbf{Z}_1^N, \mathbf{Z}_2^N, N | \Sigma) \leq H(N) + 2 \log |\mathcal{Z}| \mathbb{E}N - 1.8113k. \quad (2)$$

Section IV derives a lower bound on the conditional entropy in the above inequality. Specifically, it is shown that

$$H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma) \geq 0.75(-1 + \log 3)k \text{ bits for large } k. \quad (3)$$

Using this in inequality (2), we obtain

$$\begin{aligned} 0.75(-1 + \log 3)k &\leq H(N) + 2 \log |\mathcal{Z}| \mathbb{E}N - 1.8113k, \\ \implies \frac{k}{\mathbb{E}N \log |\mathcal{Z}|} &\leq \frac{H(N)}{\mathbb{E}N \log |\mathcal{Z}|} + \frac{2}{2.25}. \end{aligned} \quad (4)$$

Furthermore, the following claim holds (see Appendix A for a proof).

Lemma 1: For k large enough, $H(N)/\mathbb{E}N \leq \epsilon$, for any $\epsilon > 0$.

Indeed, for an arbitrary probability mass function on \mathbb{N} (set of natural numbers), this ratio can take the value 1 when N follows a geometric distribution with parameter $1/2$. However, the zero error criterion restricts the possible set of probability

mass functions for the stopping time N and yields the required upper bound for the ratio.

Lemma 2: A valid stopping time N for our network satisfies

$$\Pr(N = n) \leq \left(\frac{3}{8}\right)^k |\mathcal{Z}|^{2n} \text{ for any } n \in \mathbb{N}.$$

Proof: For a given $\sigma \in \{0, 1, 2, 3\}^k$ consider the set S of values such that $\Pr(N = n | \Sigma = \sigma) > 0$. By definition of stopping time, terminal t can recover at most $|\mathcal{Z}|^{2n}$ different values of $\hat{\Sigma}$, which by the zero error criterion, is the same as the value of Σ . Thus, if $|S| > |\mathcal{Z}|^{2n}$, there is a positive probability of error. Hence, we have,

$$\begin{aligned} \Pr(N = n) &\leq \sum_{\sigma \in S} \Pr(N = n | \Sigma = \sigma) \Pr(\Sigma = \sigma), \\ &\leq |\mathcal{Z}|^{2n} \max_{\sigma} \Pr(\Sigma = \sigma), \\ &= \left(\frac{3}{8}\right)^k |\mathcal{Z}|^{2n}. \end{aligned}$$

□

IV. LOWER BOUND ON CONDITIONAL ENTROPY

In this section we derive the lower bound on $H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma)$ as stated in inequality (3). To do this, we first note that the zero error criterion enforces a requirement that the stopped sequences $\mathbf{Z}_1^N, \mathbf{Z}_2^N$ must satisfy.

Lemma 3: For a valid (k, N) network code, let $\mathbf{z}_1^{n_1} := \phi_{(s_1, t)}(\mathbf{x}_1, \mathbf{x}_3)$ and $\mathbf{z}_1^{n'_1} := \phi_{(s_1, t)}(\mathbf{x}'_1, \mathbf{x}_3)$. Similarly define $\mathbf{z}_2^{n_2}$ and $\mathbf{z}_2^{n'_2}$. Then,

- $\mathbf{z}_1^{n_1} \neq \mathbf{z}_1^{n'_1}$ for all $\mathbf{x}_1 \neq \mathbf{x}'_1$ with $\mathbf{x}_1, \mathbf{x}'_1 \in \{0, 1\}^k$ and
- $\mathbf{z}_2^{n_2} \neq \mathbf{z}_2^{n'_2}$ for all $\mathbf{x}_2 \neq \mathbf{x}'_2$ with $\mathbf{x}_2, \mathbf{x}'_2 \in \{0, 1\}^k$.

Proof: Assume otherwise and consider the two sets of inputs $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ and $(\mathbf{x}'_1, \mathbf{x}_2, \mathbf{x}_3)$ such that $\mathbf{x}'_1 + \mathbf{x}_2 + \mathbf{x}_3 \neq \mathbf{x}_1 + \mathbf{x}_2 + \mathbf{x}_3$. One can easily see that such a set of inputs exist. Then if $\mathbf{z}_1^{n_1} = \mathbf{z}_1^{n'_1}$, the terminal t is unable to compute the arithmetic sum correctly from the corresponding stopped sequences, leading to a non zero probability of error. □

Set $L_{x,y} := 2^{x+y}$ and $M_{x,y} := 3^{x+y}$. For a natural number u , let $[u] := \{1, 2, \dots, u\}$. For a vector \mathbf{v} , index its components with a natural number i and let $v(i)$ denote its i -th component.

Claim 1: Let a particular realization σ of Σ be such that x components of it equal 1 and y components of it equal 2. For a valid (k, N) network code, the conditional entropy $H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma = \sigma)$ is minimized when the probability mass function $\Pr(\mathbf{Z}_1^N = \mathbf{z}_1^N, \mathbf{Z}_2^N = \mathbf{z}_2^N | \Sigma = \sigma)$ is positive for exactly $L_{x,y}$ distinct $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ pairs.

Proof: For a σ with x 1's and y 2's in it, there are $M_{x,y}$ $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ -tuples that result in that particular sum. Within these $M_{x,y}$ input tuples, there are $L_{x,y}$ different values of \mathbf{x}_3 . We can partition all input tuples into disjoint sets that have the arithmetic sum σ based on the value of \mathbf{x}_3 . The set with the most number of input tuples in it corresponds to a

particular value which we denote $\tilde{\mathbf{x}}_3$. One can check that, for $i = \{1, 2, \dots, k\}$

$$\tilde{\mathbf{x}}_3(i) = \begin{cases} 0, & \text{if } \sigma(i) = 0, 1 \\ 1, & \text{if } \sigma(i) = 2, 3. \end{cases}$$

From Lemma 3, we know that all input tuples for a fixed \mathbf{x}_3 must receive distinct $(\mathbf{Z}_1^N, \mathbf{Z}_2^N)$ labels. Thus for any σ , we must have atleast $L_{x,y}$ distinct $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ labels as the size of the largest partition (which is the $\tilde{\mathbf{x}}_3$ -partition) is $L_{x,y}$. These instantiations of the pair process $(\mathbf{Z}_1^N, \mathbf{Z}_2^N)$ must therefore have a positive conditional probability. Note that all the input tuples that result in a particular σ are equally likely. Hence, the conditional probability of any one particular $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ label for a given σ has to be a multiple of $1/M_{x,y}$.

In Appendix B we prove the following claim from which the result follows. Let $c > 0$ and u^* be a positive integer such that $cu^* \leq 1$. For a natural number $u \leq u^*$, let \mathcal{Q}_u be the set of probability mass functions supported on $[u]$ such that for all vectors $\mathbf{q} \in \mathcal{Q}_u$, we have that $\mathbf{q}(i) \geq c > 0$, $\forall i \in [u]$.

Claim 2: For some $m \leq u^* - 1$, let

$$\mathbf{q}_m := \arg \min_{\mathbf{q} \in \mathcal{Q}_m} H(\mathbf{q}), \text{ and } \mathbf{q}_{m+1} := \arg \min_{\mathbf{q} \in \mathcal{Q}_{m+1}} H(\mathbf{q}).$$

Then, we have $H(\mathbf{q}_m) \leq H(\mathbf{q}_{m+1})$.

This follows from the fact that entropy is a concave function and attains its minimum at an extremal point of the underlying polyhedron. Thus, using any more than $L_{x,y}$ distinct labels will increase the conditional entropy. □

We now explicitly derive the conditional entropy-minimizing distribution over $L_{x,y}$ $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ labels. Index all the $L_{x,y}$ different values of \mathbf{x}_3 that result in a particular arithmetic sum σ by a natural number i and denote them as \mathbf{x}_3^i , $i \in \{1, 2, \dots, L_{x,y}\}$. Recall that the input tuples with sum σ can be partitioned into disjoint sets based on the value of \mathbf{x}_3^i . We call the set corresponding to \mathbf{x}_3^i , the \mathbf{x}_3^i -partition. Let n_i be the size of \mathbf{x}_3^i -partition.

Claim 3: Conditioned on the particular value of σ , the set of all probability mass functions on $L_{x,y}$ valid $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ labels can be represented by a family \mathcal{P} of vectors over the reals.

$$\mathcal{P} = \left\{ \mathbf{p} \in \mathbb{R}^{L_{x,y} \times 1} : \mathbf{p} = \frac{1}{M_{x,y}} \sum_{i=1}^{L_{x,y}} \mathbf{e}_i, \mathbf{e}_i \in \mathbb{R}^{L_{x,y} \times 1} \right\} \quad (5)$$

such that \mathbf{e}_i has n_i 1's and rest as 0 for every i .

Proof: Each \mathbf{x}_3^i -partition of the $(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3)$ input tuples that have the arithmetic sum σ has n_i elements in it. Each equally-likely input tuple for this σ is assigned a particular $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ label by the encoding functions. Thus, the conditional probability distribution $\Pr(\mathbf{Z}_1^N = \mathbf{z}_1^N, \mathbf{Z}_2^N = \mathbf{z}_2^N | \Sigma = \sigma)$ is determined by how frequently the $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ label is reassigned to different input tuples that have the arithmetic sum as σ .

Define a $L_{x,y}$ -length vector \mathbf{e}_i for each \mathbf{x}_3^i -partition. The components of the vector \mathbf{e}_i denote whether a particular $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ label is assigned to an input tuple in the \mathbf{x}_3^i -partition or not. Note that by Lemma 3, a $(\mathbf{z}_1^N, \mathbf{z}_2^N)$ label can be assigned to atmost one input tuple in a particular \mathbf{x}_3^i -partition.

Thus e_i has n_i components as 1 and the rest as 0. Then the frequency of occurrence of a particular (z_1^N, z_2^N) label among all the input tuples that result in the arithmetic sum σ can be found by considering the component-wise sum $\sum_{i=1}^{L_{x,y}} e_i$. Normalizing by the total number of input tuples gives the claim. \square

Theorem 1: Let $\mathbf{1}_u$ and $\mathbf{0}_v$ denote the all-ones vector with u components and the all-zeros vector with v components respectively. Let $L_{x,y}^i := L_{x,y} - n_i$ for all $i \in [L_{x,y}]$. Then

$$\mathbf{p}^* = \frac{1}{M_{x,y}} \left(\begin{bmatrix} \mathbf{1}_{n_1} \\ \mathbf{0}_{L_{x,y}^1} \end{bmatrix} + \begin{bmatrix} \mathbf{1}_{n_2} \\ \mathbf{0}_{L_{x,y}^2} \end{bmatrix} + \dots + \begin{bmatrix} \mathbf{1}_{n_{L_{x,y}}} \\ \mathbf{0}_{L_{x,y}^{L_{x,y}}} \end{bmatrix} \right) \quad (6)$$

is a probability mass function in \mathcal{P} that minimizes the entropy on $L_{x,y}$ (z_1^N, z_2^N) labels. Moreover, any other entropy-minimizing distribution is only a permutation of \mathbf{p}^* .

Proof: It will be shown in Claim 5 that \mathbf{p}^* is an extremal point of the set $\text{conv}(\mathcal{P})$, which is the convex hull of the set \mathcal{P} . Thus, it is a potential minimizer of the concave entropy function over the convex set $\text{conv}(\mathcal{P})$. Claim 6 shows that there are no other candidate minimizers, and hence

$$H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma = \sigma) \geq \min_{\mathbf{p} \in \mathcal{P}} H(\mathbf{p}) \geq \min_{\mathbf{p} \in \text{conv}(\mathcal{P})} H(\mathbf{p}) = H(\mathbf{p}^*).$$

We refer to \mathbf{p}^* as the “clumpy” distribution. \square

Let $\mathbf{e}_i^* = [\mathbf{1}_{n_i} \ \mathbf{0}_{L_{x,y}^i}]^\top$ so that $\mathbf{p}^* = \frac{1}{M_{x,y}} \sum_{i=1}^{L_{x,y}} \mathbf{e}_i^*$. Let \mathbf{e}_i denote any binary vector of length $L_{x,y}$ such that it has exactly n_i ones. From Claim 3 any $\mathbf{p} \in \mathcal{P}$ can be expressed as $\sum_{i=1}^{L_{x,y}} \mathbf{e}_i$ for appropriate choices of vectors $\mathbf{e}_i, i \in [L_{x,y}]$.

Claim 4: Let $\mathbf{d} = \mathbf{p}^* - \mathbf{p}$ and let $\mathbf{d}(i)$ represent its i -th component. Then, $\sum_{i=1}^u \mathbf{d}(i) \geq 0$ for all $u \in [L_{x,y}]$.

Proof: We show this by considering $\mathbf{e}_j^* - \mathbf{e}_j$. Note that, for $1 \leq i \leq n_i$, we have $\mathbf{e}_j^*(i) = 1 \geq \mathbf{e}_j(i)$. This implies that for $1 \leq u \leq n_i$, we have

$$\sum_{i=1}^u \mathbf{e}_j^*(i) - \mathbf{e}_j(i) \geq 0.$$

On the other hand when $n_i < u \leq L_{x,y}$, we have

$$\sum_{i=1}^u \mathbf{e}_j^*(i) - \mathbf{e}_j(i) = n_i - \sum_{i=1}^z \mathbf{e}_j(i) \geq 0,$$

where the last inequality holds because both \mathbf{e}_j^* and \mathbf{e}_j have n_i ones. As $\mathbf{d} = \frac{1}{M_{x,y}} \sum_{j=1}^{L_{x,y}} (\mathbf{e}_j^* - \mathbf{e}_j)$, the result follows. \square

Corollary 1: If $\mathbf{p} \in \text{conv}(\mathcal{P})$ and $\mathbf{p} \neq \mathbf{p}^*$, then

- $\mathbf{p}^*(i) > \mathbf{p}(i)$ for $i = \min\{k : \mathbf{p}^*(k) \neq \mathbf{p}(k)\}$, and
- $\mathbf{p}^*(j) < \mathbf{p}(j)$ for $j = \max\{k : \mathbf{p}^*(k) \neq \mathbf{p}(k)\}$.

Proof: For $\mathbf{p} \in \mathcal{P}$, substitute $u = \min\{i : \mathbf{p}^*(i) \neq \mathbf{p}(i)\}$ in Claim 4 and note that $\mathbf{d}(i) = 0 \ \forall i < u$. For the second case, note that $\sum_{i=1}^{\max\{j : \mathbf{p}^*(j) \neq \mathbf{p}(j)\}} \mathbf{d}(i) = 0$ and substitute $u = \max\{j : \mathbf{p}^*(j) \neq \mathbf{p}(j)\} - 1$ in claim 4.

Now let $\mathbf{p} \in \text{conv}(\mathcal{P})$ such that $\mathbf{p} = \sum \mu_l \mathbf{p}_l$ where each $\mathbf{p}_l \in \mathcal{P}$ and each $\mu_l > 0$ with $\sum \mu_l = 1$. Since the corollary is true for each \mathbf{p}_l , we have that $\mathbf{p}_l(i_l) < \mathbf{p}^*(i_l)$, if i_l is the first index where \mathbf{p}_l differs from \mathbf{p}^* . Suppose $i_j := \min_l i_l$ is unique, then $\mathbf{p}_l(i_j) = \mathbf{p}^*(i_j) \ \forall l \neq j$ and $\mathbf{p}_j(i_j) < \mathbf{p}^*(i_j)$.

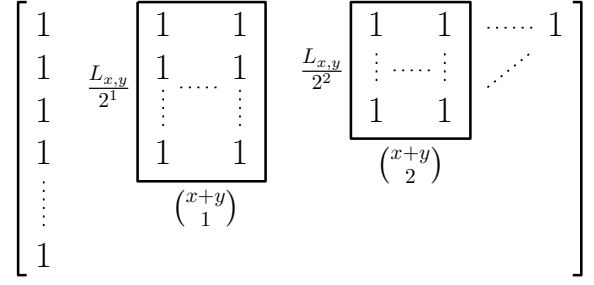


Fig. 2. Staircase structure for the $\{0,1\}$ -matrix \mathbf{A} defined in subsection IV-A. Only the 1s are shown. Boxes denote all-ones block matrices of row and column dimension as mentioned beside their length and breadth.

Hence $\mathbf{p}(i_j) = (1 - \mu_j)\mathbf{p}^*(i_j) + \mu_j\mathbf{p}_j(i_j) < \mathbf{p}^*(i_j)$. A similar argument also works when i_j is not unique. \square

Claim 5: \mathbf{p}^* is an extremal point of the convex set $\text{conv}(\mathcal{P})$.

Proof: Suppose that there exist $\mathbf{p}_1, \mathbf{p}_2 \in \text{conv}(\mathcal{P})$ such that $\mathbf{p}^* = (\mathbf{p}_1 + \mathbf{p}_2)/2$ with $\mathbf{p}_1 \neq \mathbf{p}_2$. Let i be the least index such that $\mathbf{p}_1(i) \neq \mathbf{p}_2(i)$. Then without loss of generality, $\mathbf{p}_1(i) > \mathbf{p}^*(i) > \mathbf{p}_2(i)$. But that is a contradiction to the corollary to Claim 4 and hence our claim is proved. \square

Our next claim shows that *all* extremal points of $\text{conv}(\mathcal{P})$ are permutations of the distribution \mathbf{p}^* (proof appears in Appendix C).

Claim 6: Any $\mathbf{p} \in \text{conv}(\mathcal{P})$ can be written as a convex combination of vectors which are permutations of \mathbf{p}^* .

A. Entropy of clumpy distribution

For \mathbf{p}^* as defined in (6), WLOG assume that $n_1 \geq n_2 \geq \dots \geq n_{L_{x,y}}$. Recall that n_i is the number of input tuples of the form (x_1, x_2, x_3^i) that have the arithmetic sum σ . Then $n_1 = L_{x,y}$ as it corresponds to \tilde{x}_3 -partition. If a particular x_3 differs from \tilde{x}_3 in any $1 \leq u \leq x+y$ components, then one can check that the number of input tuples with arithmetic sum σ in this particular x_3 -partition is exactly $L_{x,y}/2^u$. Also, depending on which u bits of \tilde{x}_3 are flipped, there are $\binom{x+y}{u}$ different x_3 -partitions that have $L_{x,y}/2^u$ input tuples in them. Let \mathbf{A} denote a $L_{x,y} \times L_{x,y}$ matrix with the i th column as the vector \mathbf{e}_i^* for all $i \in [L_{x,y}]$. By the above discussion, matrix \mathbf{A} has a “staircase” structure as shown in Figure 2. Each row of \mathbf{A} corresponds to a particular label and the row sum indicates its frequency of occurrence. Conditioned on σ , each input tuple that results in that sum is equally likely with probability $1/M_{x,y}$. Then the expression for the entropy of the clumpy distribution is given in equation (7). The first term in the RHS denotes the contribution to the entropy by labels which are repeated exactly once, the second term corresponds to labels repeated exactly twice, and so on.

$$H(\mathbf{p}^*) = -2^{x+y-1} \frac{\binom{x+y}{0}}{M_{x,y}} \log \frac{\binom{x+y}{0}}{M_{x,y}} - 2^{x+y-2} \frac{\binom{x+y}{0} + \binom{x+y}{1}}{M_{x,y}} \log \frac{\binom{x+y}{0} + \binom{x+y}{1}}{M_{x,y}} - \dots$$

$$-\frac{\binom{x+y}{0} + \dots + \binom{x+y}{x+y}}{M_{x,y}} \log \frac{\binom{x+y}{0} + \dots + \binom{x+y}{x+y}}{M_{x,y}} \quad (7)$$

Let the set of (z_1^N, z_2^N) labels assigned to the input tuples for a particular value of $\Sigma = \sigma$ be Z_σ . Then our discussion about the clumpy distribution implies that a lower bound for the quantity $H(Z_1^N, Z_2^N | \Sigma = \sigma)$ can be obtained by choosing $|Z_\sigma| = L_{x,y}$ and letting these labels follow the probability mass function \mathbf{p}^* . Since the network has to compute the arithmetic-sum of the messages, input tuples that result in a different σ must be provided a different (z_1^N, z_2^N) label. Thus, for two different realizations $\sigma, \tilde{\sigma}$ that have the same number of 1's and 2's, we have that $Z_\sigma \cap Z_{\tilde{\sigma}} = \emptyset$ and both $H(Z_1^N, Z_2^N | \Sigma = \sigma), H(Z_1^N, Z_2^N | \Sigma = \tilde{\sigma}) \geq H(\mathbf{p}^*)$, where x, y in the definition (equation (6)) of \mathbf{p}^* are the number of 1's, 2's respectively in σ or $\tilde{\sigma}$.

Thus a lower bound for $H(Z_1^N, Z_2^N | \Sigma) = \sum_{\sigma} \Pr(\Sigma = \sigma) H(Z_1^N, Z_2^N | \Sigma = \sigma)$ can be found by assuming that each of the conditional pmfs are clumpy. Using this, in Appendices D and E, we show the following result.

Lemma 4:

$$\frac{H(Z_1^N, Z_2^N | \Sigma)}{k} \rightarrow 0.75(-1 + \log 3) \text{ as } k \rightarrow \infty.$$

V. LOWER BOUND ON COMPUTING CAPACITY

In this section we describe a valid (k, N) network code which satisfies $k/EN = 2/2.5$ for the case when $\mathcal{Z} = \{0, 1\}$. A similar scheme can be easily extended to larger alphabets. In fact, the scheme described here is the same scheme as the one described in [2], except that no probability distribution on the inputs was used there. In what follows, all addition operations are over the real numbers.

Set k to be an even number. The encoder at s_1 computes the first $k/2$ components of the sum $\mathbf{X}_1 + \mathbf{X}_3$. Note that the components of $\mathbf{X}_1 + \mathbf{X}_3$ are iid distributed according to

$$\Pr(X_1 + X_3 = u) = \begin{cases} 1/4, & \text{if } u \in \{0, 2\} \\ 1/2, & \text{otherwise,} \end{cases}$$

and hence $H(X_1 + X_3) = 1.5$ bits. For an $\epsilon > 0$, we compress the sequence of first $k/2$ components of $\mathbf{X}_1 + \mathbf{X}_3$ based on whether they belong to the weakly typical set $A_\epsilon^{(k/2)}$ [12] or not. We can encode all the sequences in $A_\epsilon^{(k/2)}$ by using at most $\lceil \frac{k}{2}(1.5 + \epsilon) \rceil$ bits. For encoding sequences not in $A_\epsilon^{(k/2)}$, we don't need more than $\lceil \frac{k}{2} \log 3 \rceil$ bits. We add an extra bit to indicate whether the sequence being encoded belongs to the typical set or not. Having encoded the first $k/2$ bits of $\mathbf{X}_1 + \mathbf{X}_3$ in the above fashion, s_1 transmits the subsequent $k/2$ bits of \mathbf{X}_1 in an uncoded manner.

The encoder at s_2 employs a similar procedure as above except that it transmits the first $k/2$ bits of \mathbf{X}_2 in an uncoded manner and the subsequent $k/2$ bits of the component-wise sum $\mathbf{X}_2 + \mathbf{X}_3$ using typical set coding for a typical set with the same ϵ .

The terminal is able to recover the first $k/2$ components of $\mathbf{X}_1 + \mathbf{X}_3$ and \mathbf{X}_2 with zero error and the last $k/2$ components of \mathbf{X}_1 and $\mathbf{X}_2 + \mathbf{X}_3$ with zero error. From these it can correctly compute k components of the sum $\mathbf{X}_1 + \mathbf{X}_2 + \mathbf{X}_3$.

For the value of the stopping time, the terminal waits for $1 + k/2$ bits so as to obtain all the uncoded bits and the information about whether the coded bits belong to the typical set or not. Based on that, it waits for an appropriate number of bits so as to decode the required information without error. Let $\mathbf{V}_1, \mathbf{V}_2$ denote the first $k/2$ components of $\mathbf{X}_1 + \mathbf{X}_3$ and the last $k/2$ components of $\mathbf{X}_2 + \mathbf{X}_3$ respectively. Let B denote the event $\{\mathbf{V}_1 \in A_\epsilon^{(k/2)} \cap \mathbf{V}_2 \in A_\epsilon^{(k/2)}\}$. Then the expected value of the stopping time can be evaluated as follows

$$\begin{aligned} EN &= 1 + \frac{k}{2} + \Pr(B) \left\lceil \frac{k}{2}(1.5 + \epsilon) \right\rceil + (1 - \Pr(B)) \left\lceil \frac{k}{2} \log 3 \right\rceil, \\ &\leq 1 + \frac{k}{2} + \left\lceil \frac{k}{2}(1.5 + \epsilon) \right\rceil + 2\epsilon \left\lceil \frac{k}{2} \log 3 \right\rceil. \end{aligned}$$

Hence, for large k , $EN \approx \frac{5k}{4}$ and that gives our result.

VI. CONCLUSIONS AND FUTURE WORK

We have obtained new upper and lower bounds for zero error arithmetic-sum computation using variable-length network codes for a specific network. There is still a gap between the achievable rate and the upper bound. Future work will involve trying to narrow this gap. In addition, all currently known upper bounds for function computation over DAGs are based on cutsets and are recognized to be loose. It may be fruitful to examine whether the upper bound technique used in this work that operates by lower bounding the entropy of the descriptions conditional on the function value are applicable in more general scenarios.

REFERENCES

- [1] A. Giridhar and P. Kumar, "Computing and communicating functions over sensor networks," *IEEE J. Select. Areas Comm.*, vol. 23, no. 4, pp. 755–764, April 2005.
- [2] R. Appuswamy, M. Franceschetti, N. Karamchandani, and K. Zeger, "Network coding for computing: Cut-set bounds," *IEEE Trans. on Info. Th.*, vol. 57, no. 2, pp. 1015–1030, Feb 2011.
- [3] B. K. Rai and B. K. Dey, "On network coding for sum-networks," *IEEE Trans. on Info. Th.*, vol. 58, no. 1, pp. 50–63, 2012.
- [4] A. Ramamoorthy and M. Langberg, "Communicating the sum of sources over a network," *IEEE Journal on Selected Areas in Communication: Special Issue on In-network Computation: Exploring the Fundamental Limits*, vol. 31(4), pp. 655–665, 2013.
- [5] R. Ahlswede, N. Cai, S.-Y. Li, and R. W. Yeung, "Network Information Flow," *IEEE Trans. on Info. Th.*, vol. 46(4), pp. 1204–1216, 2000.
- [6] R. Koetter and M. Médard, "An Algebraic approach to network coding," *IEEE/ACM Trans. on Networking*, vol. 11, no. 5, pp. 782–795, 2003.
- [7] A. Orlitsky and J. Roche, "Coding for computing," *IEEE Trans. on Info. Th.*, vol. 47, no. 3, pp. 903–917, Mar 2001.
- [8] A. Tripathy and A. Ramamoorthy, "Sum-networks from undirected graphs: Construction and capacity analysis," in *Communication, Control, and Computing (Allerton)*, 2014 52nd Annual Allerton Conference on, Sept 2014, pp. 651–658.
- [9] —, "Capacity of sum-networks for different message alphabets," in *Information Theory (ISIT)*, 2015 IEEE International Symposium on, June 2015, pp. 606–610.
- [10] H. Kowshik and P. Kumar, "Optimal function computation in directed and undirected graphs," *IEEE Trans. on Info. Th.*, vol. 58, no. 6, pp. 3407–3418, June 2012.
- [11] C. Huang, Z. Tan, and S. Yang, "Upper bound on function computation in directed acyclic networks," in *Information Theory Workshop (ITW)*, 2015 IEEE, April 2015, pp. 1–5.
- [12] T. Cover and J. Thomas, *Elements of Information Theory*. Wiley Series, 1991.

APPENDIX A

Consider the optimization problem defined as follows, where $p_i := \Pr(N = i)$ for $i \in \mathbb{N}$ and $\Delta = \log e/\epsilon$.

$$\text{minimize} \quad \mathbb{E}N - \frac{\Delta}{\log e} H(N) = \sum p_i(i + \Delta \ln p_i) \quad (9a)$$

$$\text{subject to:} \quad p_i - \left(\frac{3}{8}\right)^k |\mathcal{Z}|^{2i} \leq 0 \text{ for all } i \in \mathbb{N}, \quad (9b)$$

$$\begin{aligned} & -p_i \leq 0 \text{ for all } i \in \mathbb{N}, * \\ & \sum p_i - 1 = 0. \end{aligned} \quad (10)$$

The Lagrangian of the objective function (9a) is

$$\begin{aligned} L(\mathbf{p}, \lambda, \boldsymbol{\nu}, \boldsymbol{\mu}) = & \sum p_i(i + \Delta \ln p_i) + \lambda(1 - \sum p_i) \\ & - \sum \nu_i p_i + \sum \mu_i \left(p_i - \left(\frac{3}{8}\right)^k |\mathcal{Z}|^{2i} \right). \end{aligned} \quad (11)$$

Here, $\lambda, \boldsymbol{\nu} \geq \mathbf{0}, \boldsymbol{\mu} \geq \mathbf{0}$ are dual variables with the natural number subscript i indexing their components. Since the Lagrangian is convex in \mathbf{p} , minimizing it involves setting

$$\begin{aligned} \frac{\partial L}{\partial p_i} = & i + \Delta \ln p_i + \Delta - \lambda - \nu_i + \mu_i = 0 \text{ for all } i, \\ \Rightarrow p_i = & \exp\left(\frac{\lambda + \nu_i - \mu_i - i - \Delta}{\Delta}\right). \end{aligned}$$

Substituting this back in equation (11), we get that the dual function is

$$\mathcal{L}(\lambda, \boldsymbol{\mu}, \boldsymbol{\nu}) = \lambda - \sum_i B(i, \lambda, \mu_i, \nu_i),$$

where

$$\begin{aligned} B(i, \lambda, \mu_i, \nu_i) := & \left[\Delta \exp\left(\frac{\lambda + \nu_i - \mu_i - i - \Delta}{\Delta}\right) \right. \\ & \left. - \mu_i \left(\frac{3}{8}\right)^k |\mathcal{Z}|^{2i} \right]. \end{aligned}$$

We evaluate the dual at a point in its domain to obtain a lower bound to the optimal value of equation (9a). For $\boldsymbol{\nu} = \mathbf{0}$ and

$$\mu_i = \begin{cases} \lfloor ck \rfloor - i, & \text{if } i \in \{1, 2, \dots, \lfloor ck \rfloor\} \\ 0, & \text{otherwise,} \end{cases}$$

where $c = \log_{|\mathcal{Z}|}(\frac{8}{3})$, the value of the dual function is

$$\begin{aligned} \mathcal{L}(\lambda) = & \lambda - \left(\frac{3}{8}\right)^k \sum_{i=1}^{\lfloor ck \rfloor} (\lfloor ck \rfloor - i) |\mathcal{Z}|^{2i} \\ & - \Delta \sum_{i=1}^{\lfloor ck \rfloor} \exp\left(\frac{\lambda - \lfloor ck \rfloor - \Delta}{\Delta}\right) - \Delta \sum_{i > \lfloor ck \rfloor} \exp\left(\frac{\lambda - i - \Delta}{\Delta}\right). \end{aligned}$$

We can separately evaluate that

$$\left(\frac{3}{8}\right)^k \sum_{i=1}^{\lfloor ck \rfloor} (\lfloor ck \rfloor - i) |\mathcal{Z}|^{2i} \leq \frac{|\mathcal{Z}|^2}{(|\mathcal{Z}|^2 - 1)^2} \text{ for large } k.$$

Using this and expanding the geometric series, we obtain that

$$\begin{aligned} \mathcal{L}(\lambda) \geq & \lambda - \frac{|\mathcal{Z}|^2}{(|\mathcal{Z}|^2 - 1)^2} - \Delta \sum_{i=1}^{\lfloor ck \rfloor} \exp\left(\frac{\lambda - \lfloor ck \rfloor - \Delta}{\Delta}\right) \\ & - \frac{\Delta}{e - e^{1-1/\Delta}} \exp\left(\frac{\lambda - \lfloor ck \rfloor}{\Delta}\right). \end{aligned}$$

If we choose $\lambda = ck/2$, we can see that the value of the dual function will be positive for large k . Hence, for any probability mass function that satisfies Lemma 2, we get that the value of $\mathbb{E}N - \frac{\Delta}{\log e} H(N) \geq 0$, i.e., $H(N)/\mathbb{E}N \leq \epsilon$. \square

APPENDIX B PROOF OF CLAIM 2

We first show that atleast one of the components of \mathbf{q}_{m+1} is exactly equal to c . Pick any $q \in \mathcal{Q}_{m+1}$ and arrange its components in nonincreasing order so that $\mathbf{q}(m+1)$ is its smallest component. If $\mathbf{q}(m+1) > c$, then we can express \mathbf{q} as a convex combination of two other elements of \mathcal{Q}_{m+1} as follows. Note that $\mathbf{q}(1) \geq \mathbf{q}(m+1) \geq c$.

$$\mathbf{q} = \frac{1}{2} \begin{bmatrix} \mathbf{q}(1) + \mathbf{q}(m+1) - c \\ \mathbf{q}(2) \\ \vdots \\ \mathbf{q}(m) \\ c \end{bmatrix} + \frac{1}{2} \begin{bmatrix} \mathbf{q}(1) - \mathbf{q}(m+1) + c \\ \mathbf{q}(2) \\ \vdots \\ \mathbf{q}(m) \\ 2\mathbf{q}(m+1) - c \end{bmatrix}$$

By concavity of entropy function we then have that this $\mathbf{q} \neq \mathbf{q}_{m+1}$. Thus \mathbf{q}_{m+1} must have atleast one component that is equal to c , and WLOG let $\mathbf{q}_{m+1} = c$. Let Q be a random variable on $[m+1]$ following the probability distribution \mathbf{q}_{m+1} . Let $I = \mathbf{1}_{\{Q=m+1\}}$ be the indicator function for the event $\{Q = m+1\}$. Then $\Pr(I = 1) = c$ and $\Pr(I = 0) = 1 - c$. Then we have that

$$\begin{aligned} H(Q, I) = & H(Q) = H(I) + H(Q|I) \\ = & H(I) + \Pr(I = 0)H(Q|I = 0). \end{aligned}$$

Since $\Pr(Q|I = 0)$ is a probability distribution over $[m]$ and $\Pr(Q = i|I = 0) > \Pr(Q = i)$ for all $i \in [m]$, $\Pr(Q|I = 0) \in \mathcal{Q}_m$. Hence we have that $H(Q|I = 0) \geq H(\mathbf{q}_m)$ and using this in the previous equation, we get

$$H(Q) \geq c \log \frac{1}{c} + (1 - c) \log \frac{1}{1 - c} + (1 - c)H(\mathbf{q}_m).$$

Since $mc < 1$, we have that $-\log c > \log m$. But $\log m$ is the entropy of the uniform distribution over $[m]$ and hence $\log m \geq H(\mathbf{q}_m)$. Using this in the previous equation we get

$$H(Q) \geq cH(\mathbf{q}_m) + (1 - c) \log \frac{1}{1 - c} + (1 - c)H(\mathbf{q}_m) \geq H(\mathbf{q}_m).$$

The same argument can be repeated to show that $H(\mathbf{q}_{m+1}) \geq H(\mathbf{q}_m) \geq H(\mathbf{q}_{m-1})$ and so on. Hence, any probability mass function that satisfies a lower bound on the values of each of its probability masses necessarily has an equal or larger entropy if it is nonzero over a larger set. \square

APPENDIX C
PROOF OF CLAIM 6

We show that any $\mathbf{p} \in \text{conv}(\mathcal{P})$ can be written as a convex combination of permuted \mathbf{p}^* 's. Since $\mathbf{p} \in \text{conv}(\mathcal{P})$, we can write that $\mathbf{p} = \sum_i \mu_i \mathbf{p}_i$ where each $\mathbf{p}_i \in \mathcal{P}$, each $\mu_i \geq 0$ and $\sum_i \mu_i = 1$. One can then see that if the claim is true for each \mathbf{p}_i then it is also true for \mathbf{p} . Hence we focus on $\mathbf{p} \in \mathcal{P}$ and show that above claim holds for it. Without loss of generality, we can assume that the target vector \mathbf{p} is arranged in non-increasing order, otherwise we permute its components so that the largest component is the first component and the successive components are in a non-increasing order. We can then reverse the permutation for every vector in its convex combination finally to get back our original vector.

Algorithm 1 returns a list of vectors, each of which can be expressed as a convex combination of permuted \mathbf{p}^* 's. Using this list, we can find the convex combination of permuted \mathbf{p}^* 's for any given $\mathbf{p} \in \mathcal{P}$ arranged in non-increasing order. The notation $\mathbf{p}'[i \leftrightarrow j]$ indicates the vector \mathbf{p}' with its values at the i th and j th components interchanged, while all the other components remain the same.

Algorithm 1 Convex combination of permuted \mathbf{p}^* 's for \mathbf{p} .

Input: $\mathbf{p} \in \text{conv}(\mathcal{P})$ arranged in nonincreasing order, \mathbf{p}^* .

Output: A list L of vectors.

- 1: Initialize $\mathbf{p}' \leftarrow \mathbf{p}^*$, $L \leftarrow \emptyset$. \mathbf{v}, λ are temporary variables.
- 2: **while** $\mathbf{p}' - \mathbf{p} \neq \mathbf{0}$ **do**
- 3: Find the smallest indices i, j such that $\mathbf{p}'(i) > \mathbf{p}(i)$ and $\mathbf{p}'(j) < \mathbf{p}(j)$.
- 4: Evaluate

$$\lambda := \frac{\min\{\mathbf{p}'(i) - \mathbf{p}(i), \mathbf{p}(j) - \mathbf{p}'(j)\}}{\mathbf{p}'(i) - \mathbf{p}'(j)}.$$

- 5: Add the vector $\mathbf{v} := (1 - \lambda)\mathbf{p}' + \lambda\mathbf{p}'[i \leftrightarrow j]$ to the list L .
 - 6: Update $\mathbf{p}' \leftarrow \mathbf{v}$.
 - 7: **end while**
-

Intuitively, the algorithm finds the difference between \mathbf{p} and \mathbf{p}^* and computes an intermediate vector \mathbf{v} that can be written as a convex combination of permuted \mathbf{p}^* 's. Following this, it finds the new difference between \mathbf{v} and the target vector \mathbf{p} and repeats the previous procedure. Finally, it stops when the intermediate vector equals \mathbf{p} . The correctness of the algorithm is ensured by the following claims.

Claim 7: Let $m := \min\{\mathbf{p}'(i) - \mathbf{p}(i), \mathbf{p}(j) - \mathbf{p}'(j)\}$. Then, at step 5 in algorithm 1,

$$\mathbf{v}(i) = \mathbf{p}'(i) - m, \text{ and } \mathbf{v}(j) = \mathbf{p}'(j) + m.$$

Also $\mathbf{v}(k) = \mathbf{p}'(k)$ for all $k \neq i, j$.

Proof: Substituting $\lambda = m/(\mathbf{p}'(i) - \mathbf{p}'(j))$ gives the result. \square

Claim 8: At step 3 in algorithm 1, $i < j$ and $\sum_{k=1}^u (\mathbf{p}'(k) - \mathbf{p}(k)) \geq 0$ for all $u \geq j$.

Proof: We prove this by induction on the iteration number of the WHILE loop. Suppose the indices i, j found at the t th iteration be i_t, j_t and the \mathbf{v} evaluated at step 5 be denoted by \mathbf{v}_t . Then by Claims 4 and 7, we conclude that $i_1 < j_1$ and $\sum_{k=1}^u (\mathbf{v}_1(k) - \mathbf{p}(k)) \geq 0$ for $u \geq j_1$. As induction hypothesis, we assume that $i_t < j_t$ and $\sum_{k=1}^u (\mathbf{v}_t(k) - \mathbf{p}(k)) \geq 0$ for $u \geq j_t$.

If at the t th iteration, $m = \mathbf{p}'(i_t) - \mathbf{p}(i_t)$, then by Claim 7, we get that $\mathbf{v}_t(i_t) - \mathbf{p}(i_t) = 0$ and $\mathbf{v}_t(j_t) - \mathbf{p}(j_t) \leq 0$. Also $\mathbf{v}_t(k) - \mathbf{p}(k) = \mathbf{p}'(k) - \mathbf{p}(k)$ for all $k \neq i_t, j_t$. This implies that at the $(t + 1)$ th iteration, $j_{t+1} = j_t$. Furthermore, note that \mathbf{v}_{t+1} and \mathbf{v}_t differ only at the indices i_{t+1} and j_{t+1} . For all $u \geq j_{t+1} = j_t$

$$\sum_{k=1}^u (\mathbf{v}_{t+1}(k) - \mathbf{p}(k)) = \sum_{k=1}^{j_t} (\mathbf{v}_t(k) - \mathbf{p}(k)) \geq 0.$$

Since $\mathbf{v}_{t+1}(j_{t+1}) = \mathbf{v}_t(j_t)$ and $\mathbf{v}_t(j_t) - \mathbf{p}(j_t) \leq 0$ it must be true that $i_{t+1} < j_{t+1}$.

On the other hand, if $m = \mathbf{p}(j_t) - \mathbf{p}'(j_t)$ at the t th iteration, then similarly $\mathbf{v}_t(j_t) - \mathbf{p}(j_t) = 0$, $\mathbf{v}_t(i_t) - \mathbf{p}(i_t) \geq 0$ and $\mathbf{v}_t(k) - \mathbf{p}(k) = \mathbf{p}'(k) - \mathbf{p}(k)$ for all $k \neq i_t, j_t$. This implies that $i_{t+1} = i_t$ and $j_{t+1} > j_t$. Since $i_{t+1} < j_{t+1}$ and these are the only two indices affected at the $(t + 1)$ th iteration, we have that for all $u \geq j_{t+1}$

$$\sum_{k=1}^u (\mathbf{v}_{t+1}(k) - \mathbf{p}(k)) = \sum_{k=1}^u (\mathbf{v}_t(k) - \mathbf{p}(k)) \geq 0$$

as $u \geq j_{t+1} > j_t$. This completes the induction step and proves our claim. \square

Claim 9: At step 4 of the algorithm, $\lambda \in [0, 1]$.

Proof: By definition of the indices i, j , we have that $\mathbf{p}'(i) > \mathbf{p}(i)$ and $\mathbf{p}(j) > \mathbf{p}'(j)$. From Claim 8, we have that $i < j$. Also, by assumption, \mathbf{p} is arranged in nonincreasing order. This implies that $\mathbf{p}(i) \geq \mathbf{p}(j)$. That gives the following string of inequalities

$$\mathbf{p}'(i) > \mathbf{p}(i) \geq \mathbf{p}(j) > \mathbf{p}'(j).$$

This implies that $\lambda \geq 0$. In addition it also implies that

$$\begin{aligned} \mathbf{p}(i) \geq \mathbf{p}'(j) &\Leftrightarrow \mathbf{p}'(i) - \mathbf{p}(i) \leq \mathbf{p}'(i) - \mathbf{p}'(j), \\ \mathbf{p}(j) \leq \mathbf{p}'(i) &\Leftrightarrow \mathbf{p}(j) - \mathbf{p}'(j) \leq \mathbf{p}'(i) - \mathbf{p}'(j). \end{aligned}$$

These imply that $\lambda \leq 1$. \square

The above claims conclude that \mathbf{v} is a convex combination of vectors from $\text{conv}(\mathcal{P})$. In the following claim we prove that \mathbf{v} finally converges to \mathbf{p} in a bounded number of steps.

Claim 10: The WHILE loop in algorithm 1 terminates after a bounded number of iterations.

Proof: From calculations in Claim 8 we concluded that at the end of the t th iteration, either $\mathbf{p}'(i_t) = \mathbf{p}(i_t)$ and/or $\mathbf{p}'(j_t) = \mathbf{p}(j_t)$ based on the value of m . Thus $\mathbf{p}' - \mathbf{p}$ will have a zero element at atleast one of i_t or j_t . Also, none of the indices for which the difference $\mathbf{p}' - \mathbf{p}$ is already zero are affected by the algorithm as the inequality at step 3 is strict. Thus, the number of zero elements in the vector $\mathbf{p}' - \mathbf{p}$

increases by atleast one in every successive iteration. Since there are a finite number of components, it finally stops when the difference is the zero vector. \square

APPENDIX D

Note that expression for entropy in equation (7) can be expressed as follows

$$H(\mathbf{p}^*) = \frac{1}{M_{x,y}} \sum_{j=1}^{x+y+1} [2^{x+y-j}] \left(\sum_{s=0}^{j-1} \binom{x+y}{s} B(x, y, j) \right) \quad (13)$$

where

$$B(x, y, j) = (x+y) \log 3 - \log \sum_{t=0}^{j-1} \binom{x+y}{t}.$$

For a lower bound, we ignore the ceiling and simplify parts of the RHS of equation (13) as follows.

$$\begin{aligned} & \sum_{j=1}^{x+y+1} 2^{-j} \sum_{t=0}^{j-1} \binom{x+y}{t} \\ &= \frac{1}{2} \binom{x+y}{0} + \frac{1}{2^2} \left[\binom{x+y}{0} + \binom{x+y}{1} \right] + \dots \\ & \quad + \frac{1}{2^{x+y+1}} \left[\binom{x+y}{0} + \dots + \binom{x+y}{x+y} \right], \\ &= \binom{x+y}{0} \left[\frac{1}{2} + \frac{1}{2^2} + \dots + \frac{1}{2^{x+y+1}} \right] \\ & \quad + \binom{x+y}{1} \left[\frac{1}{2^2} + \dots + \frac{1}{2^{x+y+1}} \right] + \dots \\ & \quad + \binom{x+y}{x+y} \frac{1}{2^{x+y+1}}, \\ &= \binom{x+y}{0} \left[\frac{1}{2^0} - \frac{1}{2^{x+y+1}} \right] + \binom{x+y}{1} \left[\frac{1}{2^1} - \frac{1}{2^{x+y+1}} \right] \\ & \quad + \dots + \binom{x+y}{x+y} \left[\frac{1}{2^{x+y}} - \frac{1}{2^{x+y+1}} \right], \\ &= \binom{x+y}{0} \frac{1}{2^0} + \binom{x+y}{1} \frac{1}{2^1} + \dots + \binom{x+y}{x+y} \frac{1}{2^{x+y}} \\ & \quad - \left[\binom{x+y}{0} + \binom{x+y}{1} + \dots + \binom{x+y}{x+y} \right] \frac{1}{2^{x+y+1}}, \\ &= \left(\frac{3}{2} \right)^{x+y} - \frac{1}{2}. \end{aligned}$$

We also have that

$$\begin{aligned} & \sum_{j=1}^{x+y+1} 2^{-j} \left(\sum_{s=0}^{j-1} \binom{x+y}{s} \right) \log \sum_{t=0}^{j-1} \binom{x+y}{t} \\ &= \frac{1}{2} \binom{x+y}{0} \log \binom{x+y}{0} \\ & \quad + \frac{1}{2^2} \left[\binom{x+y}{0} + \binom{x+y}{1} \right] \log \left[\binom{x+y}{0} + \binom{x+y}{1} \right] \\ & \quad + \dots + \frac{1}{2^{x+y+1}} \left[\sum_{s=0}^{x+y} \binom{x+y}{s} \right] \log \sum_{t=0}^{x+y} \binom{x+y}{t}, \end{aligned}$$

$$\begin{aligned} &= \binom{x+y}{0} \left(\frac{1}{2} \log \binom{x+y}{0} + \frac{1}{2^2} \log \left[\binom{x+y}{0} + \binom{x+y}{1} \right] \right. \\ & \quad \left. + \dots + \frac{1}{2^{x+y+1}} \log \left[\binom{x+y}{0} + \dots + \binom{x+y}{x+y} \right] \right) \\ & \quad + \binom{x+y}{1} \left(\frac{1}{2^2} \log \left[\binom{x+y}{0} + \binom{x+y}{1} \right] + \dots \right. \\ & \quad \left. + \frac{1}{2^{x+y+1}} \log \left[\binom{x+y}{0} + \dots + \binom{x+y}{x+y} \right] \right) \\ & \quad + \dots + \binom{x+y}{x+y} \frac{1}{2^{x+y+1}} \left[\binom{x+y}{0} + \dots + \binom{x+y}{x+y} \right], \\ &< \binom{x+y}{0} \left[\frac{1}{2} \log 2^{x+y} + \frac{1}{2^2} \log 2^{x+y} \right. \\ & \quad \left. + \dots + \frac{1}{2^{x+y+1}} \log 2^{x+y} \right] \end{aligned}$$

$$\begin{aligned} & \quad + \binom{x+y}{1} \left[\frac{1}{2^2} \log 2^{x+y} + \dots + \frac{1}{2^{x+y+1}} \log 2^{x+y} \right] \\ & \quad + \dots + \binom{x+y}{x+y} \frac{1}{2^{x+y+1}} \log 2^{x+y}, \\ &= \binom{x+y}{0} (x+y) \left[\frac{1}{2^0} - \frac{1}{2^{x+y+1}} \right] \\ & \quad + \binom{x+y}{1} (x+y) \left[\frac{1}{2^1} - \frac{1}{2^{x+y+1}} \right] + \dots \\ & \quad + \binom{x+y}{x+y} (x+y) \left[\frac{1}{2^{x+y}} - \frac{1}{2^{x+y+1}} \right], \\ &= (x+y) \left[\left(\frac{3}{2} \right)^{x+y} - \frac{1}{2} \right]. \end{aligned}$$

Putting the parts together, we get that

$$H(\mathbf{p}^*) \geq \frac{(x+y)L_{x,y}}{M_{x,y}} (-1 + \log 3) \left[\left(\frac{3}{2} \right)^{x+y} - \frac{1}{2} \right].$$

APPENDIX E

We use the bound obtained in Appendix D for the entropy of the clumpy distribution to obtain the following.

$$\begin{aligned} H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma) &= \sum_{\sigma} \Pr(\Sigma = \sigma) H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \Sigma = \sigma) \\ &\geq \sum_{\substack{x, y=0 \\ x+y \leq k}}^k \left(\frac{k! 2^{k-x-y}}{x! y! (k-x-y)!} \left(\frac{1}{8} \right)^{k-x-y} \left(\frac{3}{8} \right)^{x+y} \right. \\ & \quad \left. \times \frac{(x+y)(-1 + \log 3) L_{x,y}}{M_{x,y}} \left[\left(\frac{3}{2} \right)^{x+y} - \frac{1}{2} \right] \right), \\ &\geq \sum_{\substack{x, y=0 \\ x+y \leq k}}^k \frac{k! (\log 3 - 1) (x+y)}{x! y! (k-x-y)! 4^k} \left[\left(\frac{3}{2} \right)^{x+y} - \frac{1}{2} \right], \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{4^k} \left[\sum_{\substack{x, y = 0 \\ x + y \leq k}}^k \frac{k!(x+y)(\log 3 - 1)}{x!y!(k-x-y)!} \left(\frac{3}{2}\right)^{x+y} \right. \\
&\quad \left. - \sum_{\substack{x, y = 0 \\ x + y \leq k}}^k \frac{k!(\log 3 - 1)(x+y)}{x!y!(k-x-y)!2} \right] \\
&= \frac{\log 3 - 1}{4^k} \left[2 \cdot \frac{3}{2} \cdot k \cdot 4^{k-1} - \frac{1}{2} \cdot 2 \cdot k 3^{k-1} \right], \\
&= \frac{\log 3 - 1}{4^k} \left[\frac{3k}{4} 4^k - \frac{k}{3} 3^k \right], \\
&\implies \frac{H(\mathbf{Z}_1^N, \mathbf{Z}_2^N | \boldsymbol{\Sigma})}{k} \rightarrow 0.75(\log 3 - 1) \text{ as } k \rightarrow \infty.
\end{aligned}$$